Chiffrement symétrique : modes de chiffrement

- Chiffrement par bloc:
  - L'algo de chiffrement est capable de chiffrer des blocs de taille fixe
  - découper le message en bloc et chiffrer chaque bloc
  - Plusieurs modes opératoires peuvent être employées (voir plus loin: ECB, CBC, ...)
- Chiffrement en continu
  - On chiffre le texte en clair au fur et à mesure qu'il se présente
  - Utile dans le cadre d'application réseau
  - Une solution: utiliser la technique du masque jetable avec des masques précalculés ou calculés au vol à partir d'une clef de base

Méthode de chiffrement par bloc: problématique

- Algorithmes de chiffrement : chiffrent des blocs de taille fixe (par ex. 128 bits)
- Comment chiffrer 4Go de données ?
  - Réponse (trop) simple : on découpe le fichier en blocs de 128 bits et on chiffre chaque bloc indépendamment
  - C'est la méthode ECB
  - D'autres méthodes corrigent les défauts d'ECB: CBC, OFB, CTR, …
- Comment chiffrer des données qui ne font pas un nombre entier de blocs de 128 bits ?
  - Réponse: utilisation d'un bourrage réversible
    - ajout de données avant chiffrement
    - Suppression des données après déchiffrement

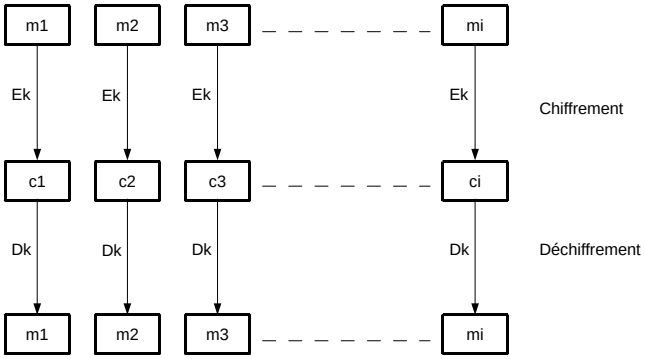
Chiffrement par bloc: ECB

- Electronic code Book (ECB) : on chiffre chaque bloc indépendamment des autres
  - Soit m le message en clair: $m=m_1 m2_{...} m_p$
  - c la version chiffrée de m: $c=c_1 c_{2...} c_p$
  - E une fonction de chiffrement capable de chiffrer des blocs, D la fonction de déchiffrement associée
  - K une clef de chiffrement pour E, déchiffrement pour D
- ECB:
  - Chiffrer: $c_1=E(m_1, k)$, $c_2=E(m_2, k)$, ..., $c_p=E(m_p, k)$
  - Déchiffrer: $m_1=D(c_1)$, $m_2=D(c_2)$, ..., $m_p=D(c_p, k)$

Chiffrement par bloc: ECB

- ECB:
  - Chiffrer: $c_1=E(m_1, k)$, $c_2=E(m_2, k)$, ..., $c_p=E(m_p, k)$
  - Déchiffrer: $m_1=D(c_1)$, $m_2=D(c_2)$, ..., $m_p=D(c_p, k)$
- On suppose que c2 est verollé: c'2. Quelle conséquence sur le déchiffrement :
  - m1=D(c1) : OK; D(c'2) ne donne **m2**
  - m3=D(c3) : OK, m4=D(c4): OK

Chiffrement par bloc: ECB



Chiffrement par bloc: ECB

- Electronic code Book (ECB) : on chiffre chaque bloc indépendamment des autres
  - Avantage:
    - Simple
    - On peut modifier un morceau des données sans modifier toutes la version chiffrée
    - On peut précalculer les versions chiffrées des blocs pour gagner en vitesse

Chiffrement par bloc: ECB

- Electronic code Book (ECB) : on chiffre chaque bloc indépendamment des autres
  - Défaut: mauvaise sécurité
    - Un bloc sera tout le temps chiffré de la même manière à l'intérieur d'un message ou dans des messages différents avec la même clef
    - L'espion peut se faire un dictionnaire s'il arrive à obtenir les versions chiffrées et en clair de certains message
  - Amélioration:
    - Ajouter un texte aléatoire avant chaque bloc à chiffrer
    - Le retirer au déchiffrement
    - Défaut: on augmenter la taille des données à chiffrer ce qui peut poser des problèmes de performances

Chiffrement par bloc: CBC

- Cypher Bloc Chaining : les blocs sont chiffrés en fonction les uns des autres.
  - En mode CBC: le texte en clair est combiné par un XOR avec la version chiffrée du bloc précédent
  - 2 messages avec le même début et chiffrés avec la même clef ont leur version chiffrée qui commence pareil :
    - Solution : On ajoute un premier bloc IV (vecteur d'initialisation).Cet IV (ou un élément permettant de le générer) est transmis en clair.
  - Avantage:
    - sûr
  - Défaut:
    - Une erreur rend illisible toute la suite des données
    - Peformance: Non parallélisable

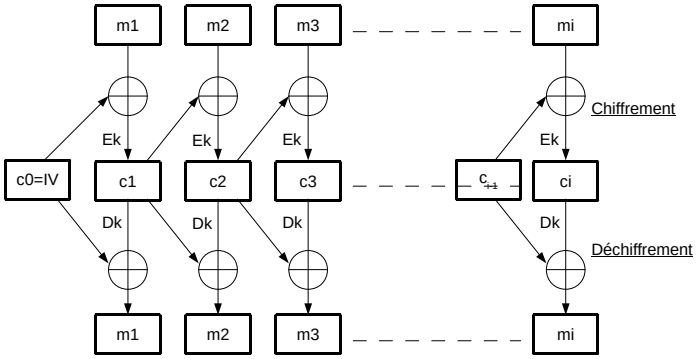
Choix du vecteur d'initialisation

- Fixe: NON car ça ne résoud pas le problème de deux fichiers commençant de la même façon.
- Compteur : VI =0 pour le message 1, VI=1 pour le suivant, ...: déconseillé (trop peu de différence entre IV pouvant annuler des différences entre messages)
- IV aléatoire: suppose l'utilisation d'un générateur aléatoire (difficile à faire) et la transmission de l'IV ce qui alourdit les petits messages
- Utilisation d'un nonce (Number used ONCE) : on utilise, par ex. un compteur comme nonce. L'IV est la version chiffrée du nonce. Le nonce est transmis en clair avec le message. Taile non << taille bloc.

Chiffrement par bloc: CBC

- CBC :
  - Chiffrer:
    - c_0=IV,
    - c_1=E($c_0 \oplus m_1$,k),
    - c_2=E($c_1 \oplus m_2$,k), ...,
    - c_p=E($c_{p-1} \oplus m_p$,k)
  - Déchiffrer:
    - c_0=IV,
    - m_1=$c_0 \oplus$D(c_1,k),
    - m_2=$c_1 \oplus$D(c_2,k), ...,
    - m_p=$c_{p-1} \oplus$D(m_p,k)

Chiffrement par bloc: CBC

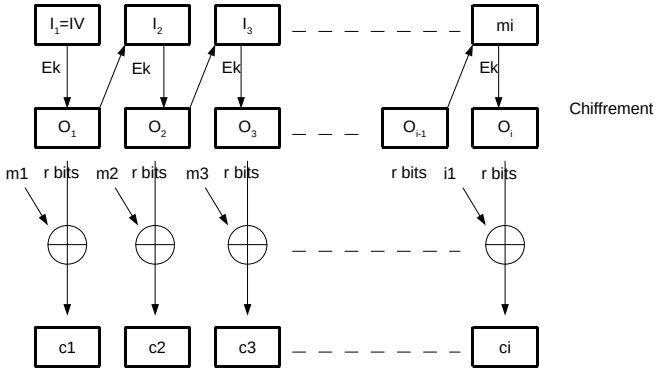


Chiffrement par bloc: OFB

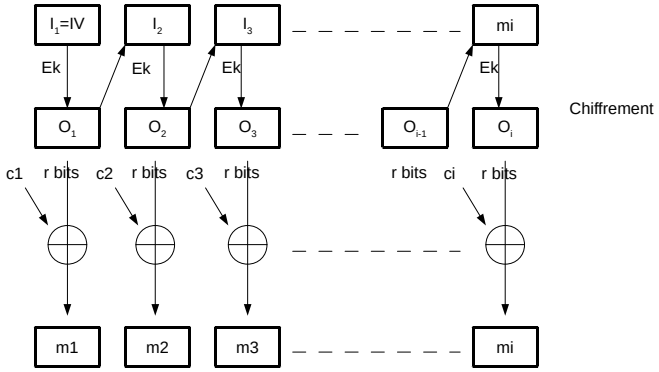
- Mode OFB (Output Feedback) ou à rétroaction de sortie
  - E: chiffrement de blocs de longueur n
  - 1 ≤ r ≤ n
  - Un vecteur d'initialisation IV
  - On découpe les données à chiffrer en blocs de taille r
- Chiffrement:
  - I_1=IV
  - O_j=E(I_j,k)
  - t_j les r premiers bits de O_j
  - c_j= $m_j \oplus t_j$
  - I_{j+1}=O_j

Déchiffrement:
I_1=IV
O_j=E(I_j,k)
t_j les r premiers bits de O_j
m_j= $c_j \oplus t_j$
I_{j+1}=O_j

Chiffrement par bloc: OFB



Déchiffrement par bloc: OFB



Même algorithme que pour le chiffrement

Chiffrement par bloc: OFB

- Avantages:
  - Simplicité
  - Dès qu'on a l'IV, on peut précalculer les O_j ce qui peut être pratique d'un point de vue performances dans certains contextes
  - Deux blocs identiques d'un même fichier seront chiffrés de façon différentes
  - En cas d'erreur, seul le bit erroné sera faux
- Défauts:
  - Le chiffrement d'un bloc en clair dépend seulement de sa position ce qui est plus faible qu'avec cbc où il dépendait du bloc chiffré précédent
  - Des cycles si $O_i=O_j$ alors $O_{i+1}=O_{j+1}$,...

Résistance aux erreurs de transmission

- Si un bloc chiffré est détérioré lors des transmissions (un méfait habituel du troll des réseaux), on montrera en TD que :
  - Méthode ECB: Un bloc déchiffré sera verollé
  - Méthode CBC: deux blocs déchiffrés seront verollés
  - Méthode OFB: Un bloc déchiffré sera verollé

Chiffrement par bloc: CTR

- Dans une version ultérieure de ce document

Chiffrement par bloc: taille des blocs

- Considérons 2 blocs tels que $C_i=C_j$
- Soit $E(k, P_i \oplus C_{i-1})=E(k, P_j \oplus C_{j-1})$
- Donc $P_i \oplus C_{i-1}=P_j \oplus C_{j-1}$
- Donc $P_i \oplus P_j=C_{i-1} \oplus C_{j-1}$
- On en déduit donc des informations sur le texte en clair.
- Il est donc important d'éviter que $C_i=C_j$.
- Cela impose que le nombre de valeurs possible des C_i ($=2^n$ avec n = taille des blocs) soit suffisamment grand par rapport à la taille des messages

Chiffrement par bloc: taille des blocs

- Paradoxe des anniversaires (Richard von Mises):
  - Combien de personnes doit-on rassembler pour avoir plus d'une chance sur deux que deux personnes de ce groupe aient leur anniversaire le même jour de l'année.
  - Réponse: 23, ce qui choque un peu l'intuition.
  - À partir d'un groupe de 57 personnes, la probabilité est supérieure à 99 %

Chiffrement par bloc: taille des blocs

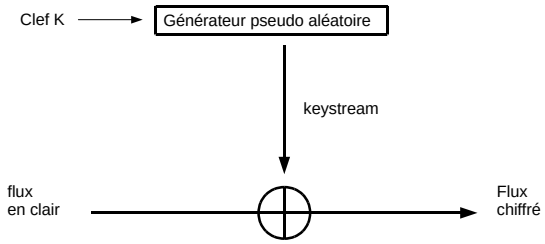
- Produire des blocs chiffrés identiques ouvre la porte à certaines attaques (voir plus haut)
- Avec des blocs de 64 bits, il faut 2^{32} (# 32Go) blocs distincts pour trouver une collision avec une chance sur deux
  - De nos jours, c'est trop peu
  - Si le mode opératoire est faible (mauvais aléa), la quantité nécessaire peut baisser
- AES utilise des blocs de 128 bits
  - Il faut alors 2^{64} blocs distincts, soit 256 exaoctets pour avoir une chance sur deux de trouver une collision

Chiffrement en continu ou par flot

- Les données sont traitées en flux
  - On commence à chiffrer
    - Sans avoir lu l'intégralité du message
    - Sans avoir sa longueur
- Exemples de chiffrement par flot
  - Masque jetable
  - RC4 (SSL, WiFi)
  - E0/1 (Bluetooth)
  - A5/1, A5/2, A5/3 (GSM)

Chiffrement en continu ou par flot

- On combine
  - Un peuso aléa (keystream ou « flux de clef »)
  - Avec le flux de données
- La combinaison est souvent un simple XOR



Chiffrement par flot

- Les chiffrements par flot sont
  - Très rapides
  - Adaptés aux application temps réel
- Problèmes
  - Propagation d'erreur en cas de problème de synchronisation
  - La sécurité repose sur la qualité du générateur pseudo aléatoire
  - Sécurité non prouvable

Masque jetable

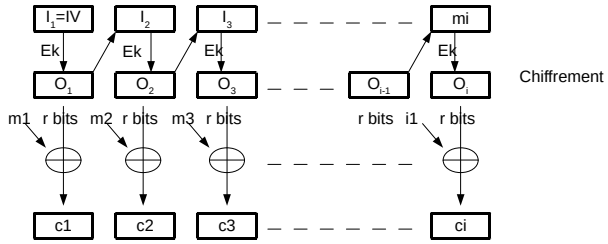
- Masque jetable:
  - La clef est une suite aléatoire parfaite
  - La clef est aussi longue que le message
  - La clef n'est jamais réutilisés
  - La sûreté est prouvée
- Chiffrement par flot
  - S'inspire du masque jetable
  - Mais
    - pseudo aléa généré à partir d'une valeur clef
    - Problème de réutilisation de la suite pseudo aléatoire si la clef ne change pas
      - => utilisation d'un valeur d'initialisation IV

Chiffrement par flot: pseudo aléa

- Il doit être impossible à disintiguer d'un véritable aléa
  - Equilibre: probabilité de ½ d'avoir 0 ou 1 en sortie
  - Indépendance: chaque bit de sortie est indépendant de ce qui précède et de ce qui suit
- Difficile à satisfaire
  - De nombreuses failles de grande ampleur ont eu comme source un mauvais aléa (ssl debian, netscape, ...)
- Exemple:
  - 414213; 732050; 236067; 645751
  - La suite paraît raisonnable mais c'est int(sqrt(2i+1))

Chiffrement par flot

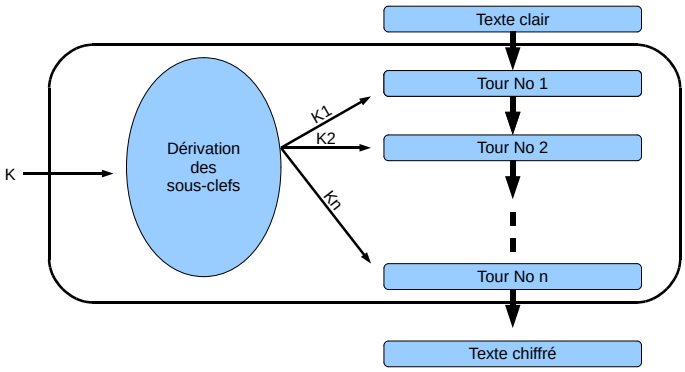
- Certains modes opératoires des chiffrements par blocs permettent leur utilisation comme chiffrement par flot. Exemple: OFB



Construction des chiffrements par bloc

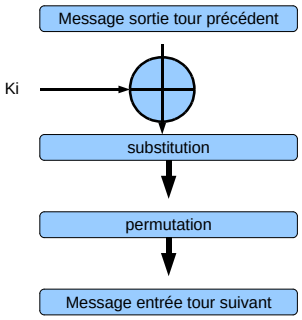
- Idée: itérer suffisamment de fois des opérations simples ***judicieusement choisies*** permet d'avoir une excellente sécurité
- Le chiffrement est une succession d'étapes similaires (appelées « tour » ou itération) utilisant chacun une sous-clef
  - Construction itérative et modulaire
  - Les sous-clefs sont dérivées de la clef secrète
  - Les fonctions utilisées par un tour doivent être optimisées et sont en général des opérations simples

Construction des chiffrements par bloc



Méthode par substitutions/permutations

- Décomposition d'un tour:
  - Utilisation de la clef Ki
  - Fonction de substitution
  - Fonction de permutation

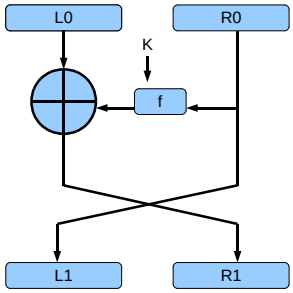


Substitution: un symbole du texte clair est remplacé par un autre symbole. Cf tables de substitution du DES (S-Boxes). Ajoute de la confusion.
permutation: on échange entre eux les symboles du texte en clair. Ajoute de la diffusion

Schéma de Feistel

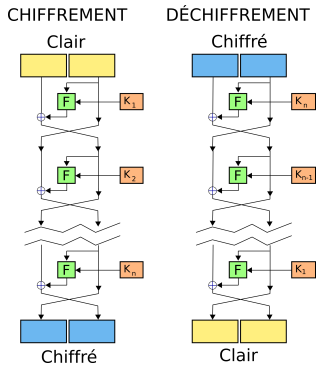
- Chiffrement:
  - L1=R0
  - R1=L0 ⊕ f(R0, K)
- Déchiffrement:
  - R0=L1
  - L0=R1⊕f(R0, K)

La fonction f est appelée fonction de confusion



Réseau de Feistel

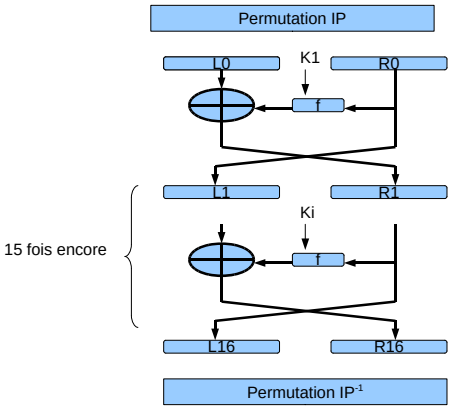
- Constitué de plusieurs tours (ou rondes) ou étages de Feistel
- Chiffrement et déchiffrement sont efficaces si chaque tour l'est
- La complexité du décryptage croit de façon exponentielle avec le nombre de tour



DES: Data Encryption Standard

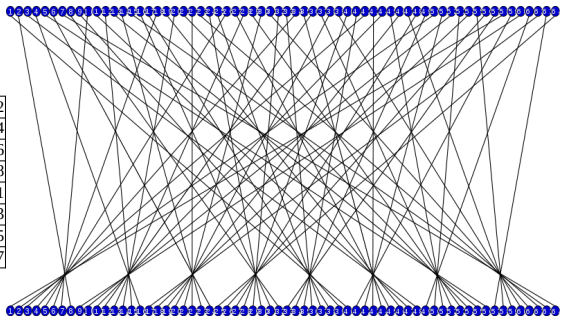
- Propriétés:
  - Algorithme à clef privée
  - Chiffrement par blocs de 64 bits
  - Feistel 16 tours avec des sous-clefs de 48 bits
  - Conçu dans les années 1970
  - Considéré comme obsolète de nos jours
- Conception:
  - À partir de l'algorithme Lucifer (IBM, 1970)
  - Corrigé par la NSA :
    - Clefs de 56 bits au lieu de 64
    - Boîtes S corrigées
    - Les but de la NSA ne seront découverts qu'en 1990
  - Standard FIPS 46-2 en 1977

DES: schéma



DES: permutation initiale IP

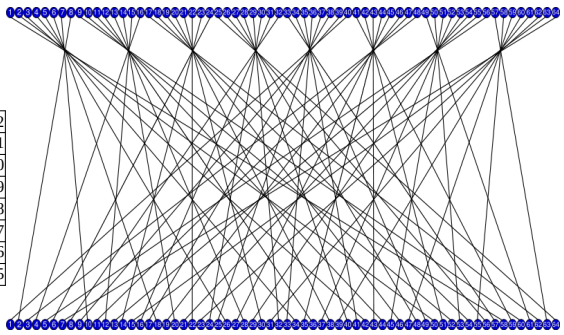
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7



Le 58e bit devient le premier, le 50e bit devient le deuxième, ..le 1er bit devient le 40e
si m=p1p2...p64 alors P(m)=p58p50...p7

DES: permutation finale

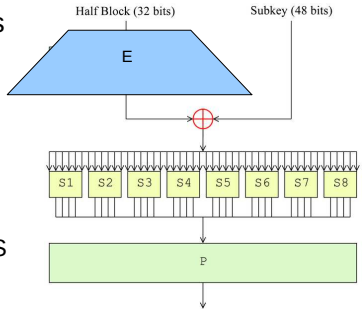
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25



Le 40e bit devient le premier, le 8e bit devient le deuxième, ...le deuxième bit devient le 50e, ...
cette permutation est l'inverse de la première.

DES: fonction f

- La fonction f est une fonction de 32 bits vers 32 bits constituée:
  - d'une expansion de Ri de 32 bits vers 48 bits
  - d'un XOR avec 48 bits dérivés de la clef
  - de l'application de 8 sous-fonctionsde 6 bits vers 4 bits : les boîtes S
  - d'une permutation des 32 bits sortants

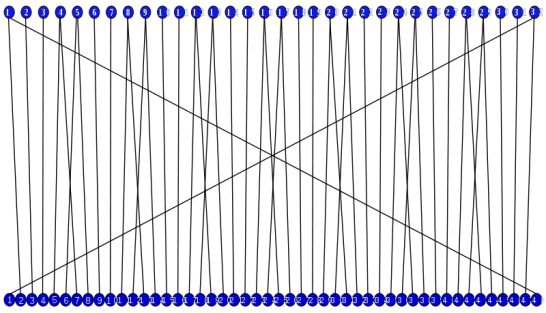


DES: fonction f

- La fonction f n'est pas inversible (elle n'a pas besoin de l'être)
- L'expansion duplique certains bits
- Les boîtes assurent la non linéarité (6 bits -> 4 bits pour chaque boîte)
- La permutation renforce la diffusion (tout comme le fait le croisement du schéma de Feistel)

DES: expansion E

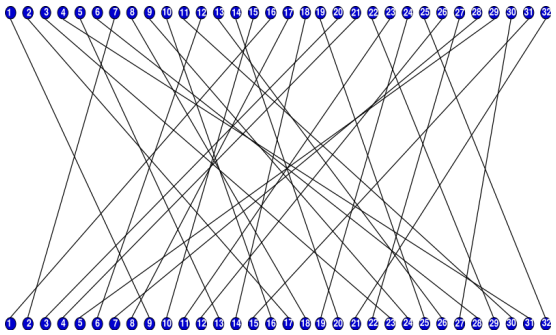
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1



Le 32e bit devient le premier, le 1er bit devient le 2e, ...

DES: permutation P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25



Le 16e bit devient le 1er, le 7e bit devient le 2e, ...

DES: boîtes de substitutions

Utilisation des boîtes-S:
si B=b1b2b3b4b5b6
s(B) est le mot binaire situé à la ligne b1b6 et à la colonne b2b3b4b5.
La numérotation des lignes et des colonnes commence à 0.
Exemple: S1(111001):
- Ligne 11=ligne3
- Colonne 1100=colonne 12
- S1(111001)=10=1010

S1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

DES: boîtes de substitutions

S5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

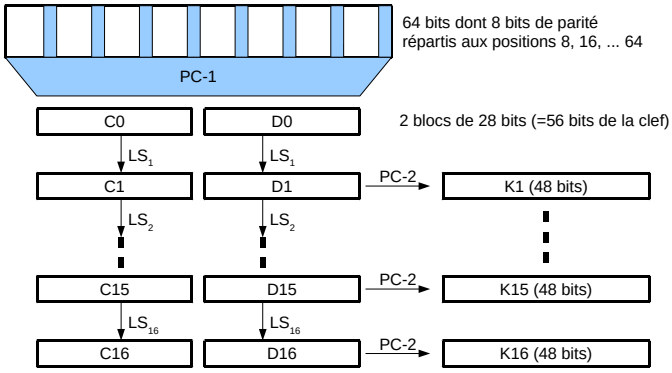
S8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

DES: génération des clefs des rondes

- Décalage: LSI : 28 bits -> 28 bits
  - LS_i(C) décalage vers la gauche de V_i bits
  - V_i=1 si i =1,2,9 ou 16
  - V_i=2 sinon
- PC1: une fonction qui retourne 2 mots de 28 bits à partir d'un mot de 64 bits
  - Elle supprime les bits de parité (8, 16, ..., 64)
  - On obtient la clef de 56 bits en 2 mots de 28 bits
- PC2:
  - 28 bitsx28bits -> 48 bits
  - Son résultat fait 48 bits et sert de paramètre à la fonction f de DES

DES : génération des clefs

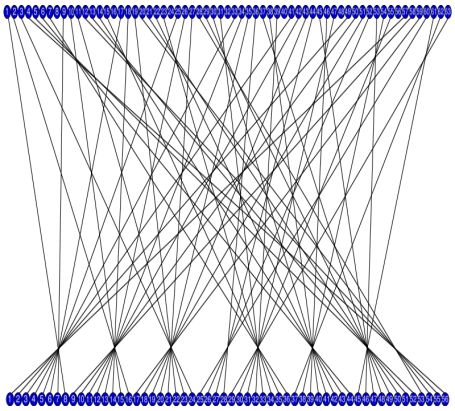


DES: génération des clefs des rondes PC1

Gauche						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
Droite						
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

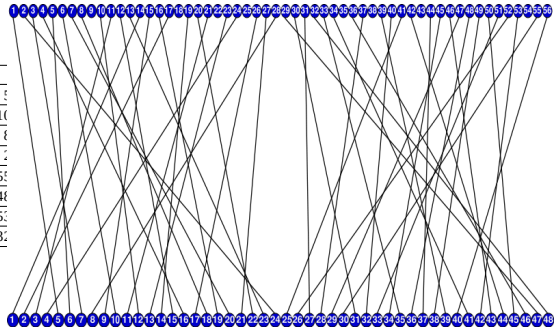
Si $k=k_1k_2...k_{64}$ alors :

- $C=k_{57}k_{49}...k_{36}$
- $D=k_{63}k_{55}...k_4$



DES: génération des clefs des rondes PC2

PC-2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32



$PC2(b_1b_2...b_{56})=b_{14}b_{17}...b_{32}$

DES: performances

- Des opérations simples:
  - La structure de Feistel est rapide
  - La dérivation des clefs est particulièrement simple et efficace
  - Les boîtes S se cablent facilement en matériel
- DES utilise des opérations simples facilement implémentables sur des puces spécialisées
  - XXXMb/s sur un XXX à XXX MHz
  - 1,5Gb/s sur une puce spécialisée bas de gamme
-

DES: sécurité

- Après 16 tours, résultat statistiquement plat
  - Aucune propriété du message source ne sera détectable
- Une légère modification du texte chiffré ou de la clef => de grosses modifications du texte chiffré
  - Bonné résistance aux attaques par analyse différentielles

DES: cryptanalyse

- Force brute : espace des clefs : 2^{56}
  - À la portée d'une PME (cf TD)
  - En 1998, l'EFF (Electronic Frontier Fondation) construit une machine
    - Craquant DES en 9 jours
    - Coût: $250 000
- Taille des blocs:
  - 64 est devenu court
  - On peut obtenir deux blocs chiffrés identiques ce qui laisse la porte ouverte à certaines attaques

DES: amélioration

- Idée d'amélioration :
  - Combiner plusieurs chiffrement DES de suite avec des clefs différentes
  - Le résultat n'est pas un DES
    - E(k1, E(k2, E(k3, M)) n'est pas égal à E(k4, M) (avec E: chiffrement DES et C: déchiffrement DES)
    - Sinon, pas d'augmentation de la complexité
  - Plusieurs tentatives :
    - Double DES : complexité #64 bits
    - Triple DES-EEE : 3 chiffrements avec des clefs différentes: E(k1, E(k2, E(k3, M))
    - Triple DES-EDE: E(k1, D(k2, E(k1, M)))
  - 3 fois plus lent que DES

DES: double DES

- Double DES:
  - E(k1, E(K2, M))
  - Atteint-on le niveau de sécurité d'un espace de clef de 56+56=112 bits. NON

Double DES: Attaque

1)On suppose que l'on connait un coupe clair M/chiffré C (un bloc d'un long message suffit)
2)On chiffre M avec toutes les clefs possibles : 2^{56}clefs possible
3)On trie les clefs : $56*2^{56}$ (n log n)
4)On déchiffre C avec toutes les clefs possible : 2^{56}clefs possible
- On cherche la version déchiffrée avec la clef j dans l'ensemble de l'étape 2: au pire en log n donc au pire $56*2^{56}$
- Si on trouve i une clef telle que E(i,M)=D(j,C) alors, on a trouvé le bon couple de clefs

- Complexité totale équivalente à celle d'un espace de clef de 64 bits

Double DES: Attaque

- L'attaque précédente ne permet pas d'avoir la certitude de trouver le bon couple de clef
  - On a 2^{64} messages chiffrés possibles
  - On a 2^{112} clefs possibles
  - Donc en moyenne, il y a $2^{112}/2^{64} = 2^{48}$couples de clefs permettant de transformer M en C
  - Donc si on trouve un couple qui marche, c'est l'un des couples qui marchent mais pas forcément le couple de clefs.
- Si on a deux textes clair M/ Chiffré C, le nombre moyen de clefs permettant de passer de M en C pour les deux textes est 2^{-16}. (16=2*64-112)
  - Donc si on trouve un couple qui marche, c'est probablement le bon couple de clef

Triple DES:

- De nombreuses variantes possible
  - Certaines ont des faiblesses
  - DES-EEE : 3 clefs de 56 bits => 168 bits mais l'attaque « meet in the middle » rend la sécurité équivalente à un espace de clef de 112 bits
- La variante conseillée (FIPS 46-3) est DES-EDE:
  - 2 clefs
  - E(k1, D(k2, E(k1, M)))
- Espace de clef de 112 bits
  - Mais en tenant compte des attaques possibles, équivalent à un espace de clef de 80 bits
- Triple Des a été très utilisé avant l'arrivé d'AES pour palier les faiblesses de DES